Keijo Karvinen

DYNAAMINEN PDF-DOKUMENTTI WEB SERVICE -TEKNIIKOIN

Opinnäytetyö
Tietojenkäsittely

Toukokuu 2010

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 24.5.2010
Tekijä(t) Keijo Karvinen	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittely

Nimeke

Dynaaminen PDF-dokumentti Web Service –tekniikoin

Tiivistelmä

Tämä opinnäytetyö käsittelee PDF-dokumentin mahdollisuuksia ja Web Services -tekniikoita. PDF-tiedostomuodosta on viime vuosina tullut lähes käytännön standardi sähköiseen muotoon tallennettaville lomakkeille. Niitä käyttävät mm. Kela ja Verohallinto. PDF-dokumenttia voidaan selata Internetissä, tulostaa paperille, sitä voidaan kommentoida ja jakaa verkossa.

Toimeksiantajana toimivan Puulan kalastusalueen tavoitteena oli saada helppokäyttöinen karttasovellus, josta voidaan katsoa kalastusalueen vesialueet omistavien osakaskuntien tietoja. Pohjana työllä toimivat PDF-dokumentit, joille Maanmittauslaitos on piirtänyt kartat.

Opinnäytetyöni tutkimusongelmana oli löytää tapa yhdistää PDF-dokumetit tietokantaan Web Service -tekniikoin ja näin muodostaa dynaaminen PDF-dokumentti. Teoriaosassa käyn läpi perusteet, joiden avulla tämä on mahdollista. Toteutusosassa esittelen, kuinka olen karttasovelluksen lopulta toteuttanut. Sovelluksen ensimmäinen versio on jo käytössä.

Asiasanat (avainsanat)

Ohjelmointi, PDF, JavaScript, Web Services

Sivumäärä	**Kieli**	**URN**
26 s.	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Janne Turunen	**Opinnäytetyön toimeksiantaja** Puulan kalastusalue

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 24 May 2010
Author(s) Keijo Karvinen	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

A dynamic PDF document with the Web Services techniques

Abstract

This bachelor's thesis dealt with the possibilities of PDF document and Web Services techniques. The PDF file format has become almost a standard for storing forms in electronic format, used for example in Kela (The Social Insurance Institution of Finland) and Verohallinto (Tax Administration). PDF documents can be browsed on the internet, printed on paper, commented and shared online.

Puulan kalastusalue (Puula fishing area) had a need to obtain an easy-to-use map application where one could find information about the owners of the fishing waters. The basis for this work was PDF documents with maps drawn by Maanmittauslaitos (National Land Survey of Finland).

My research problem was to find a way to combine these PDF documents to a database with the Web Services techniques, and thus form a dynamic PDF document. The theory part introduced how this could be done. The practical part presented how I eventually implemented the map application. The first version is already in use.

Subject headings, (keywords)

Programming, PDF, JavaScript, Web Services

Pages	**Language**	**URN**
26 p.	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Janne Turunen	Puulan kalastusalue

SISÄLTÖ

1 JOHDANTO

Tämä opinnäytetyö käsittelee maksullisen Adobe Acrobat -ohjelmiston tekniikoita PDF-dokumenttien muokkaamiseen ja luomiseen sekä Web Services -tekniikoita. Monissa yhteyksissä Web Services -tekniikoista käytetään eri nimiä, kuten Web Services, Web-palvelut taikka verkkopalvelut, mutta oikeampi nimi näille tekniikoille, joita hyödyntävä toiset sovellukset, on Web Services. Suomen kielessä verkkopalveluilla tarkoitetaan ihmisten käyttämiä verkkopalveluita

Tutkimusongelmana opinnäytetyöllä on, selvittää kuinka luodaan dynaamisia PDF-dokumentteja ja onko se mahdollista toteutuksen vaatimalla tavalla. Dynaamisella tarkoitetaan mahdollista vaihtuvaa sisältöä eri käyntikertojen välillä PDF-dokumentissa. Tässä opinnäytetyössä PDF-dokumentin dynaamisuus toteutetaan Web Servicen kautta.

Toimeksiantajana toteutusosassa esitellyllä sovelluksella on Puulan kalastusalue. Heillä oli tarve saada kalastusalueen vesialueet omistavien osakaskuntien tiedot näkymään PDF-dokumenteissa, jotka toimivat karttoina. Toteuttamassani sovelluksessa PDF-dokumentin sisältämä JavaScript luo automaattisesti muistilaput tiettyyn kohtaan dokumenttia ja tiedot niihin päivittyvät Web Servicen kautta MySQL-tietokannasta. PDF-dokumentit toimivat siis karttoina ja esittävät Puulan kalastusalueen jaettuna viiteen osa-alueeseen. Näitä alueita ovat itä, länsi, pohjoinen, keski ja etelä. Puulan kalastusalueella on noin kaksisataa vesialueiden osakaskuntaa.

Opinnäytetyön teoriaosa (luvut 1-2) pitää sisällään kaksi pääkohtaa. Ensimmäisenä esittelen PDF-dokumentin ja siihen liittyvät mahdollisuudet. Toisessa osassa esittelen Web Services -tekniikoita. Toteutuksessa esittelen toteuttamani sovelluksen ja selvitän, miten olen päässyt kyseiseen lopputulokseen.

2 PDF

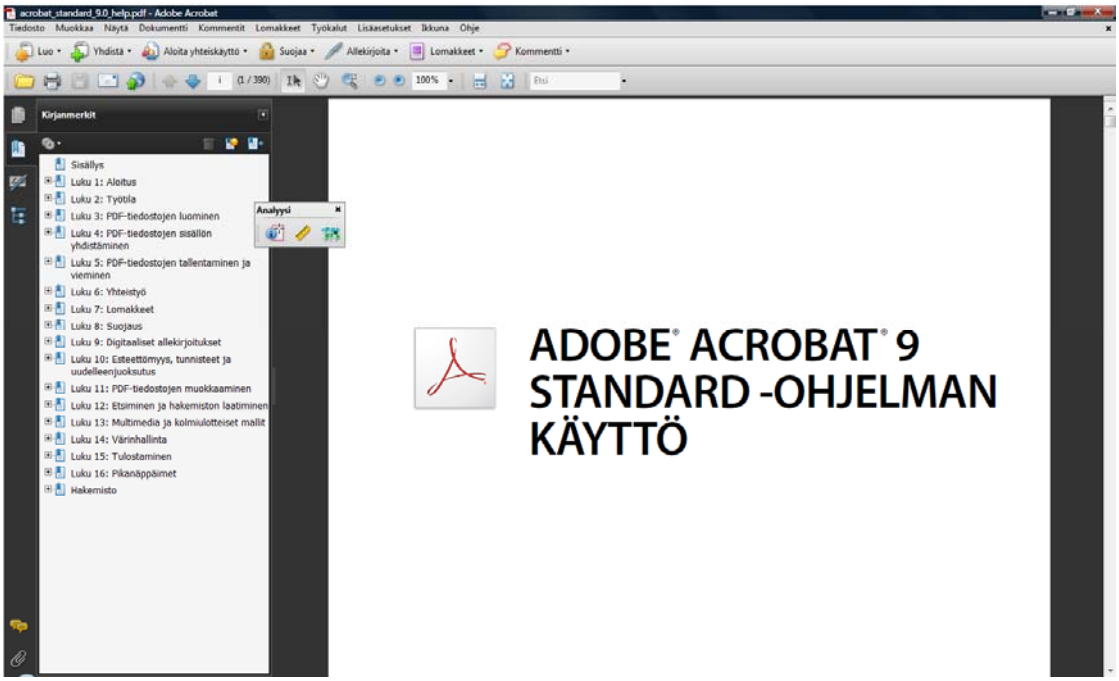
Grafiikka- ja julkaisuohjelmien valmistaja Adobe kehitti 1980-luvulla PostScript-tekniikan, jonka avulla julkaisujen valmistaminen mikrotietokoneilla (DTP, Desktop Publishing) sai osaltaan alkunsa. PostScript eli PS-sivunkuvauskielen etuja ovat resoluutio- ja käyttöjärjestelmävapaus. PostScript-tekniikan tuoman kokemuksen ja koodin perusteella Adobe loi Acrobat-tekniikan. Acrobatin ensimmäinen versio esiteltiin vuonna 1993. Acrobat-dokumentin tiedostomuotona on PDF(Portable Document Format), joka on siis sukua PostScript tiedostoille. Acrobat- tekniikka sisältää tällä hetkellä sekä Adoben omia tuotteita Acrobat-dokumenttien luomiseen, katseluun ja jalostamiseen että muiden valmistajien lisäosia toimintojen laajentamiseen. (Tarvainen 2006, 2-3.)

PDF on suunniteltu lähtökohtaisesti paperitulostusmuodoksi. Tämän johdosta PDF saattaa aiheuttaa melko suuria ongelmia niille, jotka esimerkiksi näkövamman takia tarvitsevat suurta kirjasinlajia tai puhesynteesin kautta tapahtuvaa ääniesitystä. (Korpela 2006.) PDF-dokumentin muoto on myös huono lähtökohta Internetin hakujärjestelmien kannalta, koska niiden toiminta perustuu dokumenttien tekstisisällön poimimiseen erilleen. PDF-dokumentin muoto ei yleisesti sovi ainoaksi muodoksi, jossa tieto laitetaan saataville Internetiin. Lähes aina PDF-muotoisen tiedoston nimi loppuu merkkeihin .pdf mm. siksi, että Internet Explorer tunnistaa tiedoston PDF-muotoiseksi tämän perusteella. PDF-tiedostoille varattu mediatyyppi (MIME-tyyppi) on application/pdf. (Korpela 2006.)

2.1 PDF- dokumentin ominaisuuksia ja mahdollisuuksia

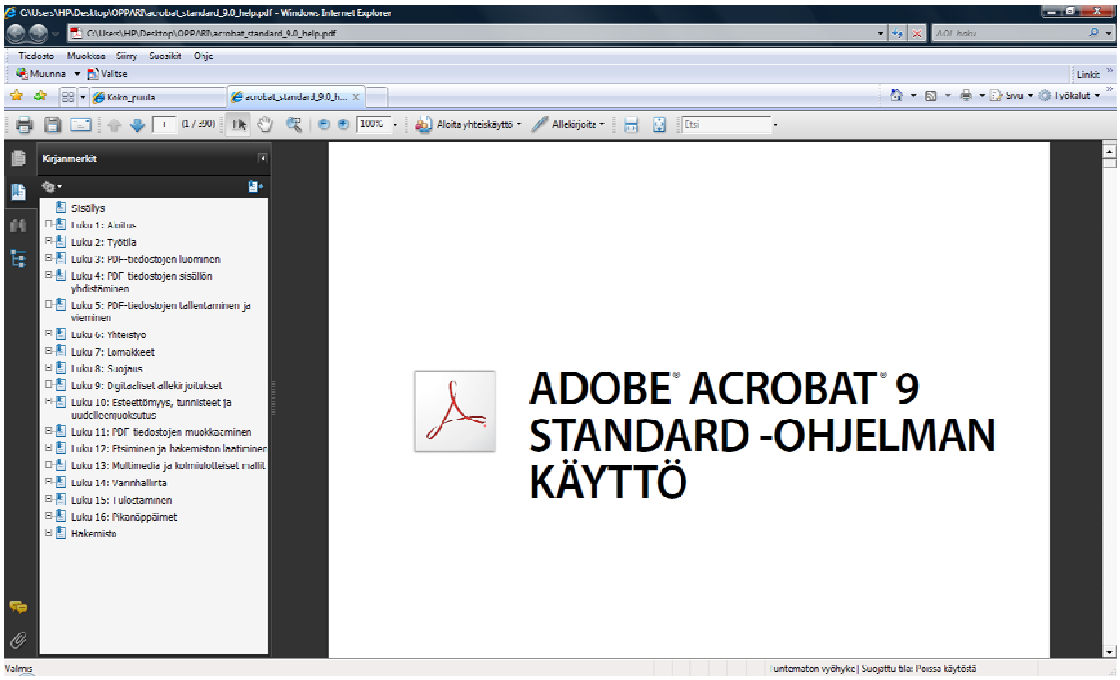
PDF - dokumentteja voi avata ja tarkastella mm. Adoben ohjelmistoilla. Adobe Acrobat on maksullinen ohjelmisto ja siitä on tällä hetkellä saatavilla Standard, Pro ja Pro Extended -paketit. Tämän opinnäytetyön kirjoitushetkellä on Acrobatista saatavilla versio 9.0.

Adobe Acrobat ja Adobe Reader avautuvat itsenäisinä sovelluksina ja Adobe Reader avautuu myös Web-selaimessa. Työskentelyalueet eroavat toisistaan ulkonaisesti vähän mutta toiminnoiltaan merkittävästi. Acrobatin erillisen sovelluksen työskentelytila sisältää dokumenttiruudun ja suunnistusruudun. Adobe PDF -tiedosto näkyy dokumenttiruudussa kuvassa 1. Vasemman puolen suunnistusruudun avulla voi selata PDF-tiedostoa. Ikkunan yläreunassa olevista työkaluriveistä löytyy työkaluja, joita voi käyttää PDF-tiedostojen parissa työskenneltäessä.(Acrobat Standard 9.0 Help, 2009.)



KUVA 1. Adobe Acrobat –ohjelmisto

Työkalurivi suunnistusruutu ja dokumenttiruutu ovat käytettävissä, kun PDF-tiedosto avataan selaimessa. Adobe Reader voidaan ladata ilmaiseksi Adoben verkkosivuilta ja tämän johdosta PDF-dokumentit soveltuvat hyvin yleiseen käyttöön ja jakeluun. Adobe Reader tulee Adobe Acrobat -ohjelmiston mukana automaattisesti.



KUVA 2. PDF- dokumentti Adobe Reader –ohjelmistossa selaimella avattuna

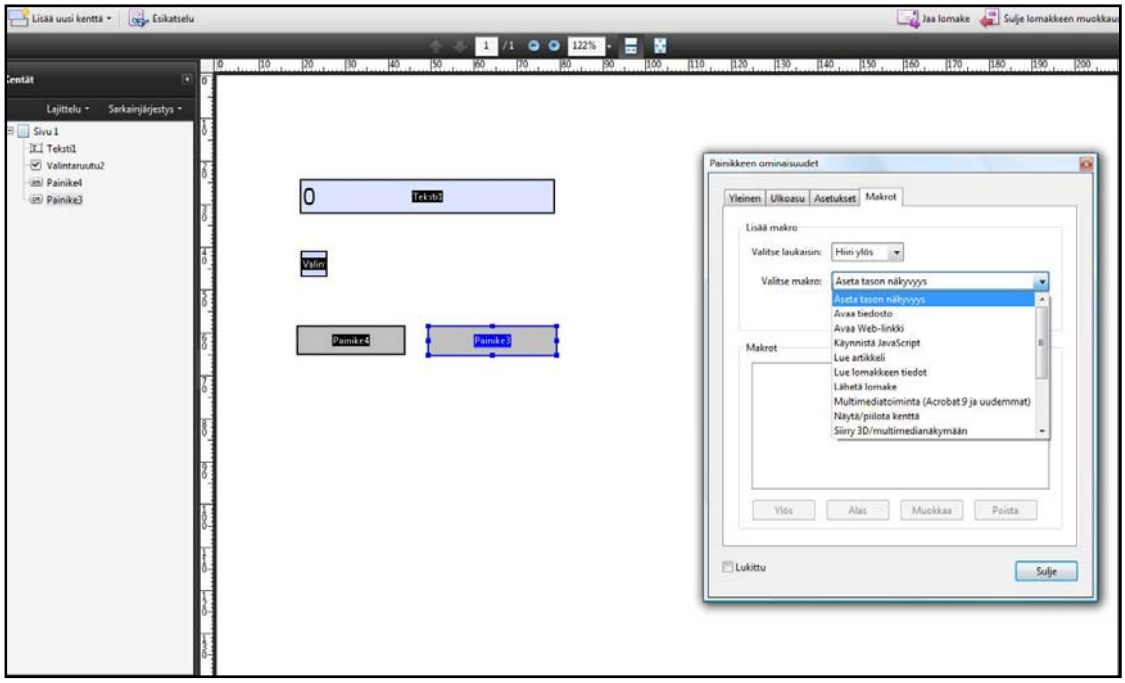
PDF-dokumentti on mahdollista myös upottaa osaksi HTML-dokumenttia niin, että se näkyy erillisessä alueessa sen sisällä. Selainten tuki eri upotusmenetelmille (iframe, embed ja object) on melko kirjava. Joka tapauksessa on yleensä käytännöllisintä laittaa upotuksen yhteyteen linkki kyseiseen PDF-dokumenttiin, jolloin käyttäjä voi hyödyntää linkkiä, jos upotus ei toimi kunnolla tai ollenkaan.(Korpela 2006.)

2.1.1 Lomakkeet, napit ja linkit

Acrobatin ohjatun lomakkeen muodostustoiminnon kautta voi luoda lomakkeita joko muuntamalla paperisia lomakkeita sähköiseksi skannaamalla tai avaamalla lomakkeena jo olemassa olevia sähköisiä tiedostoja (Excel, Word). Tietenkin lomakkeita voi itse luoda erillisissä lomakkeen luonti tilassa Acrobatissa lomakkeen teko työkaluilla, joiden ominaisuuksia ovat tekstikenttä, valintaruutu, valintanappi, luetteloruutu, yhdistelmäruutu, painike, digitaalinen allekirjoitus ja viivakoodi. Kaikkia työkalujen ominaisuuksia on mahdollista muokata ja niihin voi lisätä makroja. (Acrobat Standard 9.0 Help, 2009.)

PDF-tiedostomuodosta on viime vuosina tullut lähes käytännön standardi sähköiseen muotoon tallennettaville lomakkeille. Niitä käyttävät mm. Kela ja Verohallinto. PDF-lomake onkin omimmillaan Internetissä tai lähiverkossa. Yleensä se on tarkoitettu

täytettäväksi suoraan verkon kautta, mutta pienen tiedostokoon ansiosta se voidaan myös suunnitella ladattavaksi ensin käyttäjän omalle tietokoneelle. (Tarvainen 2006, 132.)



KUVA 3. Acrobatin lomakkeenmuokkausikkuna

Acrobatin työkaluissa on muokkauksen lisäasetukset, josta saa työkalurivin, jonka avulla voi PDF-dokumenttiin suoraan luoda nappeja, linkkejä ja tekstialueita. Nappeihin ja linkkeihin voi lisätä makroja samalla tavalla kuin kaikkiin lomakkeen ominaisuuksiin lomakkeen luonnin yhteydessä. Mahdollisia makroja ovat avaa tiedostoja, käynnistä JavaScript, lue artikkeli, lue lomakkeen tietoja, lähetä lomake, tyhjennä lomake sekä toista multimedia tai ääni. On mahdollista suorittaa myös valikkokomentoja. Valikkokomentojen avulla voi hallita PDF-dokumenttien toimintoja, esimerkiksi voi tehdä napin ja laittaa sen suorittamaan valikkokomentona näyttämään PDF-dokumentin tasot sivupaneelissa (kts. Tasot). (Acrobat Standard 9.0 Help, 2009.)

2.1.2 Kommentit ja yhteistyö

Kommentteja voi luoda Adobe Acrobatilla kommentointityökaluilla PDF-dokumentteihin. Kommentointityökalut ovat käytössä myös Adobe Readerissa, jos dokumenttiin on sallittu kommentointi. Kommentit ovat merkintöjä ja piirustuksia, joilla voi viestiä ideoista ja antaa palautetta PDF-tiedostoista. Kommenttityökaluilla voi kirjoittaa tekstiviestin tarralapputyökalulla tai piirtää piirustustyökalulla viivan, ympyrän tai muun kuvion ja sitten kirjoittaa viestin ponnahdusikkunaan. Tekstin muokkaustyökaluilla (kuva 4) voi lisätä lähdedokumenttiin muokkausmerkintöjä, joilla voi ilmaista, mitä muutoksia halutaan tehtävän. (Acrobat Standard 9.0 Help, 2009.)

KUVA 4. Acrobatin kommentti- ja merkintätyökalut (Acrobat Standard 9.0 Help, 2009)

PDF-dokumentteja voi lähettää muille tarkistetavaksi ja kommentoitavaksi suoraan Adobe Acrobat -ohjelmasta. Tarkistajat voivat lisätä kommenttinsa PDF-tiedostoon kommentointi- ja merkintätyökaluja käyttämällä. Jaetuissa tarkistuksissa tarkistajat voivat julkaista kommenttejaan jaetussa työtilassa sekä tarkastella muiden tarkistajien kommentteja ja vastata niihin. Jotta tarkistuksiin voisi osallistua, täytyy ensin luoda Acrobat.comin-verkkopalveluun oma käyttäjätili. Acrobat.com-palvelua käyttämällä voidaan lähettää useimpia dokumenttityyppejä ja jakaa työpöydällä olevia PDF-tiedostoja online-kokousten aikana. (Acrobat Standard 9.0 Help, 2009.)

2.1.3 Tasot

PDF-dokumentissa tietoja voidaan sijoittaa eri tasoille. PDF-dokumentissa voidaan tarkastella, selata ja tulostaa tasosisältöä. Tasojen näkyvyyttä ohjataan oletustilan ja aloitustilan asetuksilla. Tasojen avulla voidaan mm. piilottaa tekijänoikeusilmoituksen sisältävä tason aina, kun dokumentti näytetään näytössä, ja kuitenkin samalla varmistaa, että taso tulostettaessa näkyy aina. (Acrobat Standard 9.0 Help, 2009.)

PDF-dokumentissa käytettävät tasot perustuvat alkuperäisen sovelluksen luomiin tasoihin. Sovelluksilla, kuten InDesign, AutoCAD ja Visio voidaan luoda tasoja PDF-dokumenttiin. Erillisiin tasoihin liittyvää sisältöä voidaan tarkastella ja näyttää tai piilottaa käyttämällä tasot-paneelia. Lukituilla tasoilla olevia tietoja ei voi piilottaa. Tasot voi järjestää sisäkkäisiksi ryhmiksi päätason kanssa ja jotkin tasot voivat olla ryhmissä, joilla ei ole päätasoa. Lukitus-kuvake tasot-välilehdellä osoittaa, että sisältö on vain luettavaksi. Lukitun tason näkyvyyttä ei voi muuttaa. (Acrobat Standard 9.0 Help, 2009.)

Monitasoisesta PDF-dokumentista voidaan valita tai kopioida sisältöä valintatyökalulla (Reader-ohjelmassa PDF-tiedoston pitää sisältää käyttöoikeudet). Acrobat-ohjelmistossa sisältöä voidaan muokata kohennustyökalulla. Nämä työkalut valitsevat kaiken näkyvän sisällön riippumatta siitä, sijaitseeko se valitulla tasolla. Jos muokattu tai poistettu sisältö on liitetty yhteen tasoon, muutos näkyy kyseisellä tasolla Acrobatissa. Jos muokattu tai poistettu sisältö on liitetty useaan tasoon, muutos näkyy näillä kaikilla tasoilla. Jos esimerkiksi halutaan muuttaa otsikkoa ja kirjoittajan nimeä, jotka ovat dokumentin ensimmäisen sivun samalla rivillä mutta kahdella näkyvällä tasolla, toisella tasolla tehdyt muutokset vaikuttavat kumpaankin tasoon. Tasoihin voidaan lisätä tarkistuskommentteja, leimoja tai lomakekenttiä monitasoiseen dokumenttiin aivan samoin kuin mihin muuhun PDF-dokumenttiin tahansa. Sisältöä ei kuitenkaan lisätä tietylle tasolle, vaikka se olisikin valittu sisältöä lisättäessä. Sen sijaan sisältö lisätään koko dokumenttiin. (Acrobat Standard 9.0 Help, 2009.)

2.2 JavaScript ja PDF

HTML tarjoaa syntaksin, jota käytetään dokumenttien kuvaamiseen ja muokkaamiseen. Sen kompakti koko ja yleinen yksinkertaisuus mahdollistivat sen tulon verkkojulkaisemisen standardiksi ja ehkä laajimmin käytetyksi tekniikaksi maailmassa (Erl 2004, 19-20.)

JavaScriptin avulla ohjelmoijalla on mahdollisuus luoda “lennossa” muokattuja HTML-sivuja, joiden sisältö riippuu käyttäjän suorittamista toiminnoista. JavaScript hallitsee selaimen toimintoja, joten sen avulla voidaan avata uusia ikkunoita, esittää varoituslaatikoita ja sijoittaa räätälöityjä viestejä selainikkunan tilapalkkiin. JavaScript on asiakaspuolen ohjelmointikieli. Tämä tarkoittaa sitä, että kieli on suunniteltu

toimimaan käyttäjän tietokoneella, ei palvelimella. Tämän vuoksi JavaScriptillä on joitain sisäänrakennettuja rajoituksia, jotka johtuvat pääasiassa turvallisuussyistä. (Smith & Negrino 2007, 6-7).

```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <script type="text/javascript">
                alert("Heippa Maailma !!!");
                </script>
        </body>
</html>
```

KUVA 5. JavaScript-koodi HTML-koodiin sijoitettuna

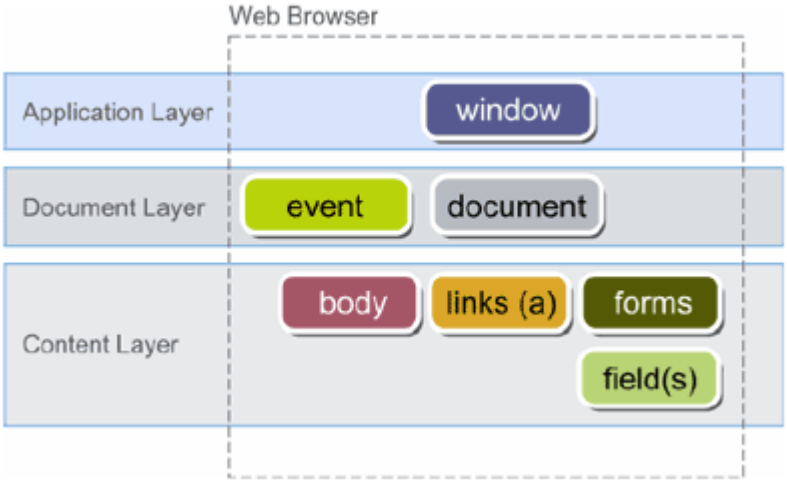
JavaScript Acrobatia varten on laajennus JavaScript-versiosta 1.5, ISO-16262, joka tunnettiin aiemmin nimellä ECMAScript. Acrobat laajentaa JavaScriptiä lisäämällä siihen uusia olioita ja niiden mukana uusia menetelmiä ja ominaisuuksia. Näiden Acrobatin omien olioiden avulla sovelluskehittäjä voi hallita asiakirjojen turvallisuutta, olla yhteydessä tietokantaan, käsitellä liitetiedostoja ja käsitellä PDF-tiedostoa niin, että se käyttäytyy kuten interaktiivinen WWW-pohjainen lomake. (Developing Acrobat Applications Using JavaScript, 2006.)

```
app.alert("Heippa Maailma!!!");
```

KUVA 6. JavaScript-koodi Acrobatissa

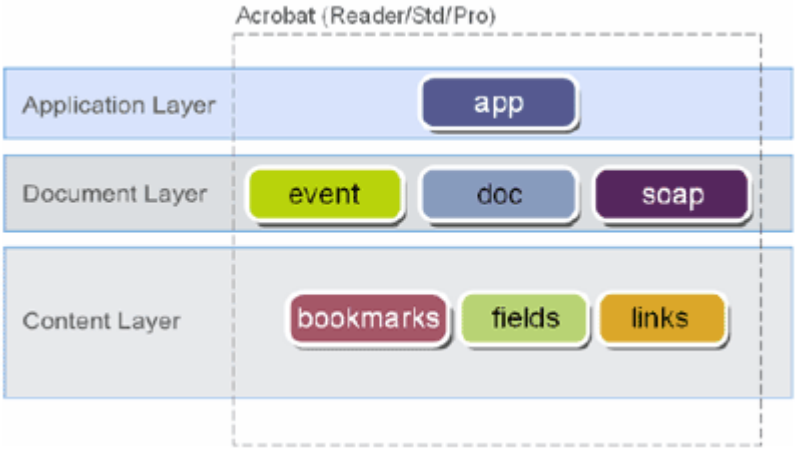
2.2.1 JavaScriptin sijoitus ja käyttö

Verkkosivu koostuu olioista, jotka on esitetty puurakenteena. JavaScript esittää dokumenttipuun alkiot olioina ja siten on mahdollista käyttää JavaScriptiä kyseisten olioiden tilan muokkaamiseen. Tällaista dokumentin olioiden esitystapaa kutsutaan dokumentin oliomalliksi (Document Object Model) tai lyhyemmin DOMiksi. (Smith & Negrino 2007, 13). Kuvassa 7 DOM on kuvattuna JavaScriptin näkökulmasta. Kuva selventää, mihin Javascriptiä on sijoitettava, jotta halutut toiminnot saataisiin käyttöön.



KUVA 7. DOM (Wraight 2004)

JavaScript selaimessa ja JavaScript PDF-dokumentissa jakavat saman yleisen kielen perustan, mutta ne toimivat ne täysin eri ohjelmointirajapinnoilla (Wraight 2004). Kuvassa 8 on kuvattu Acrobatin ohjelmointirajapinta (ADOM eli Acrobat Document Object Model) JavaScriptin näkökulmasta. Kuva selventää, mihin Javascriptiä on sijoitettava, jotta halutut toiminnot saataisiin käyttöön



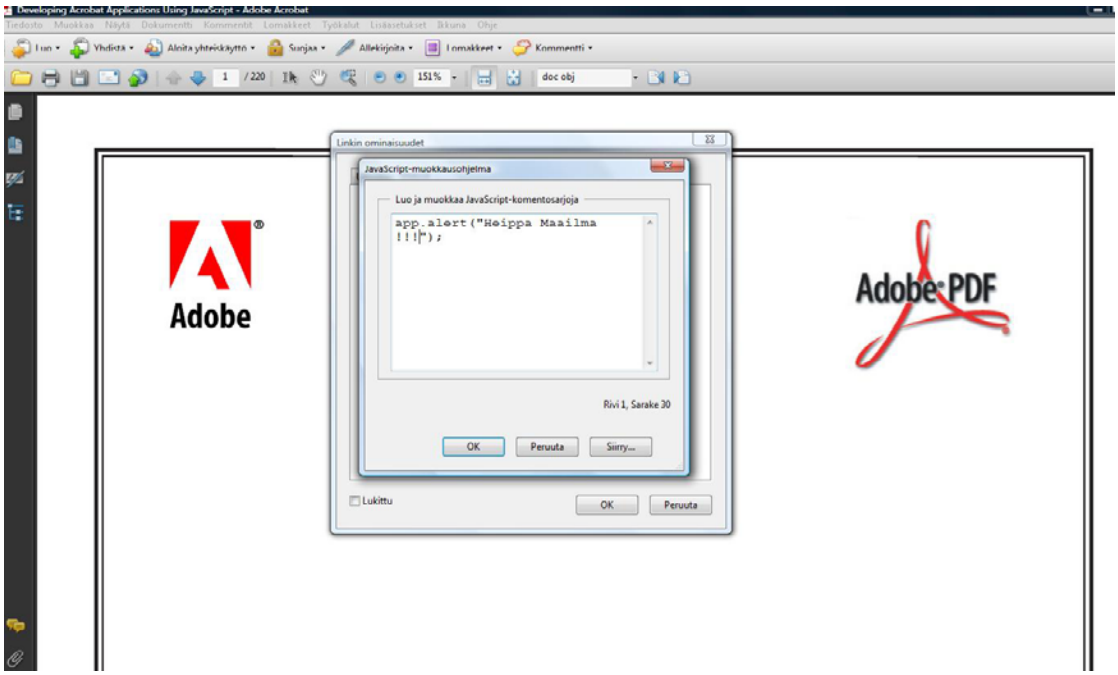
KUVA 8. ADOM (Wraight 2004)

JavaScriptiä voidaan sijoittaa eri tasoille Acrobatissa. JavaScriptiä tulee osata sijoittaa oikeille tasoille, jotta ne toimivat halutulla tavalla. JavaScriptiä voidaan lisätä seuraaville tasoille:

- sovelluksen tasolle
- dokumentin tasolle
- sivun tasolle
- kentän/Lomakkeiden tasolle
- kirjanmerkkien ja linkkien tasolla (Wraight 2004.)

Sovellustasolle sijoitettava JavaScript-koodi voi olla yleisesti hyödyllistä Acrobatille, ja se on nähtävissä kaikissa asiakirjoissa. Sovellustasolla koodit sijoitetaan erillisinä JavaScript-tiedostoina Acrobatiin. Nämä koodit ajetaan kun Acrobat avataan. Sovellustason koodien tekemiseen käytetään erillistä editoria. Editoria ei saa käynnistettyä Acrobatista sovellustason koodien luomiseen. (Developing Acrobat Applications Using JavaScript, 2006.)

Dokumenttitasolla koodit ovat muuttujia ja funktioita, jotka ovat yleensä hyödyllisiä tiettyyn dokumenttiin, mutta eivät ole käyttökelpoisia dokumentin ulkopuolella. Dokumenttitason koodit ajetaan sen jälkeen kun asiakirja on avattu. Dokumenttitason JavaScript tallennetaan itse PDF-dokumenttiin. Sivutason JavaScript suoritetaan kun tietty sivu on joko suljettu tai avattu. Kenttätasolla JavaScript liittyy tai liitetään Acrobat lomake-kenttään. Kenttätason tapahtumat tapahtuvat kun käyttäjä on vuorovaikutuksissa kentän kanssa. Kenttätason koodeja tyypillisesti toteutetaan kun halutaan validoida, muotoilla tai laskea lomake kenttien arvoja. Kuten dokumenttitason koodit, kenttätason koodit tallennetaan myös PDF-dokumenttiin. (Developing Acrobat Applications Using JavaScript, 2006.)



KUVA 9. JavaScript editori Acrobatissa

2.2.2 JavaScriptin mahdollisuudet PDF-dokumentissa

Toistuvat tehtävät ovat yleisin syy käyttää JavaScriptiä. Sen avulla voi automatisoida tehtäviä, joita et itse kerta toisensa jälkeen halua suorittaa. JavaScriptiä voidaan käyttää myös lisäämään interaktiivisuutta PDF-dokumenttiin, esimerkiksi on mahdollista luoda mukautettuja ponnahdusvalikosta joihin pääsee, kun klikkaa hiiren oikeaa näppäintä. Jos on luonut PDF-lomakkeita ja haluaa varmistaa että käyttäjät syöttävät vain kelvollista tietoa, niin JavaScriptiä voi käyttää vahvistamaan ja valvomaan syötettyjä tietoja. (Wraight 2004.)

Acrobat 7.0 ja uudemmat tarjoavat tuen SOAP 1.1 ja 1.2 standardeille, jotta PDF-lomakkeet voivat kommunikoida Web Services –tekniikoiden kanssa. Tämä tuki on mahdollistanut sen, että viestit voivat sisältää molemmat, sekä SOAP-otsikon (header) ja kehon (body) tietoa sekä sen, että voidaan lähettää binääridataa tehokkaammin ja helpottaa salattua viestintää. Kaikki tämä tekee mahdolliseksi sen, että PDF-tiedostoja voidaan yhdistää olemassa oleviin työnkulkuihin yhdistämällä XML-lomakkeita kaavioihin, tietokantoihin ja Web Services -tekniikoihin. Nämä työnkulut sisältävät mahdollisuuden jakaa kommentteja etänä tai kutsua Web Serviceä lomakkeen kenttien kautta. Kun muodostetaan yhteyttä Web Serviceen, niin JavaScript voi käyttää WSDL välityspalvelinta muodostamaan SOAP-kirjekuoren (Envelope). (Developing Acrobat Applications Using JavaScript, 2006.)

Acrobatin JavaScript tarjoaa ODBC-yhteensopivan oliomallin nimeltään Acrobat Database Connectivity (ADBC), jota voidaan käyttää dokumenttitason JavaScripteissä luomaan yhteys tietokantaan. Yhteyden avulla voidaan lisätä uutta tietoa, päivittää olemassa olevaa tietoa, ja poistaa tietokannan merkintöjä. Jotta ADBC voidaan ottaa käyttöön, on pakollista, että ODBC on asennettu työasemaan, jossa on Microsoft Windows -käyttöjärjestelmä. (Developing Acrobat Applications Using JavaScript, 2006.)

3 WEB SERVICES

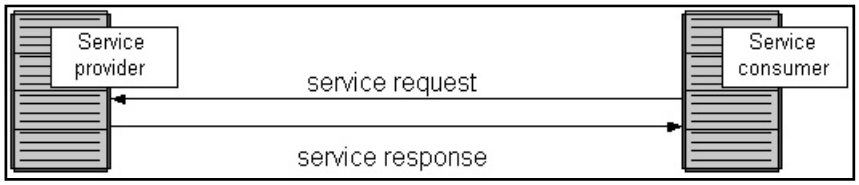
Web Services -tekniikoilla tarkoitetaan hajautettuja, ohjelmallisesti käytettäviä palveluita, joita voidaan ajatella olio-ohjelmointimallin komponentteina. Avoimiin standardeihin perustuvat Web Service -tekniikat vapauttavat sovellusohjelmat paikan, ohjelmointikielen ja erilaisten laitteistojen aiheuttamista rajoituksista. Web Services -tekniikoiden keskeisiin periaatteisiin kuuluu, että Web Services -tekniikoita hyödyntävät nimenomaan sovellukset eivät käyttäjät (Järvinen 2002, 1- 2).

Web Service on siis tapa, jonka avulla erityyppiset, eri tekniikoilla toteutetut ohjelmistot kykenevät välittämään tietoa keskenään. Integraatioarkkitehtuureja käsitellessä kyseessä on rajapintakerrokseen kuuluva tekniikka. Tähtisen mukaan Web Services -tekniikat perustuvat yksinkertaisiin XML-sanomiin, jotka välitetään yleisimmin WWW:ssä hyväksi koetun HTTP(S)-protokollan välityksellä. Muutkin siirtotiet ovat kuitenkin mahdollisia. (Tähtinen 2005, 119.)

3.1 Palvelukeskeinen arkkitehtuuri (SOA) ja Web Services

Tyypillisesti SOA toteutetaan laajalti hyväksyttyjen ja käyttöönotettujen Web Service -tekniikoiden avulla, jolloin esimerkiksi J2EE- ja Microsoft.Net-alustoille rakennetut ohjelmistot kykenevät ongelmitta keskustelemaan toistensa kanssa. SOA ei kuitenkaan ole sama asia kuin Web Service, vaan arkkitehtuurimalli, jonka avulla ohjelmistojen välistä tiedonsiirtoa voidaan yksinkertaistaa. (Tähtinen 2005, 145.)

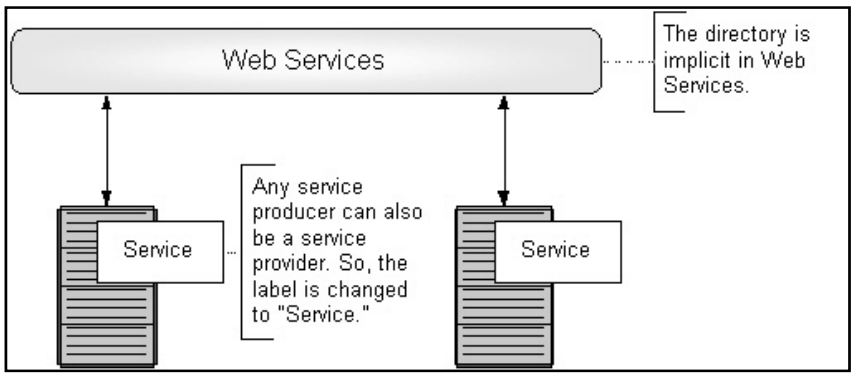
Kuvassa 10 näkyy yksinkertainen kaavio palvelukeskeisestä arkkitehtuurista. Se osoittaa palvelun käyttäjän oikealla lähettämässä palvelupyyntöviestin palveluntarjoajalle vasemmalla. Palveluntarjoaja (Service provider) palauttaa vastausviestin palvelun käyttäjälle (Service consumer). Pyyntöyhteys ja vastausyhteys määritellään jollain tavalla, joka on ymmärrettävä niin palvelun käyttäjälle kuin palvelujen tarjoajalle. Palvelun tarjoaja voi olla myös palvelun käyttäjä. (Barry 2010.)



KUVA 10. Service Oriented Architecture (Barry 2010)

Kun integroidaan Web Service -tekniikoita, niin valinta menetelmän käytöstä jää suunnittelijalle. Suurin osa Web Serviceistä käyttää joko REST- tai SOAP -tekniikoita, mutta ei molempia. (Reinheimer 2006, 98.) Termi Web Service ei siis edellytä käyttämään SOAP:ia pakkausmuotona tai käsittelymallina. Se ei myöskään edellytä käyttämään WSDL:ää palvelunkuvauskielenä. Tälläkin hetkellä on ja tulevaisuudessakin tulee olemaan paljon verkkopalveluita, jotka käyttävät raaka HTTP:tä tiedonsiirron protokollana ja jotain XML:n kaltaista dokumenttia yhteisesti sovittuna viestin sisältönä. Ehkä pitkällä aikavälillä muut tekniikat tulevat syrjäyttämään SOAPin ja WSDL:n käytön. (Booth 2003.)

Yksinkertaistettu Web Services -tekniikoiden esitystapa on hahmotettu kuvassa 11. Web Services -tekniikoiden voidaan ajatella toimivan kuten PC:n väylän, johon voidaan liittää erilaisia piirilevyjä. Palveluarkkitehtuureissa palveluntarjoaja voi olla myös palvelun käyttäjä. Tämän vuoksi kuvassa 11 lukee Web Services -väylän alla Service eikä "Service provider" ja "Service consumer". (Barry 2010.)



KUVA 11. Yksinkertainen Web Services –malli (Barry 2010)

W3C, Web Services Interoperability Organization (WS-I) sekä lukuisat teknologiatoimittajat pitävät huolen Web Service -tekniikoiden voimakkaasta kehityksestä lähivuosien ajan. Web Servicet täydentyvät lähitulevaisuudessa standardeilla, joiden avulla palvelurajapintoihin voidaan lisätä teknologiatoimittajasta riippumattomat tietoturva- ja taatun viestinvälityksen ominaisuudet. (Tähtinen 2005, 200.)

3.2 XML

Extensible Markup Language 1.0 (XML 1.0) on World Wide Web Consortiumin virallinen suositus yleiskäyttöiseksi tekstimuotoisten dokumenttien merkintäkieleksi. XML 1.0 määrittelee XML-dokumenttien luokan kieliopin sekä tarjoaa tyyppimäärityskielen, jonka avulla dokumenttien loogista rakennetta voidaan mallintaa (ja esim. erottaa kirjeet, runot ja bussiaikataulut niiden rakenteen perusteella toisistaan). Keskeisenä ideana on yhteisesti sopia, kuinka tekstimuotoista tietoa kannattaa tallettaa tietorakenteina dokumenteiksi siten, että tieto on helposti ja yksinkäsitteisesti luettavissa. XML 1.0:n ensisijainen käyttö on toimia kehittyneempien XML-sovellusten teknisenä perustana. (Nykänen, 2006, 2.)

```
<?xml version="1.0" encoding="UTF-8"?>
<esimerkki>
 <viesti>
  Heippa Maailma!
 </viesti>
</esimerkki>
```

KUVA 12. Yksinkertainen XML-dokumentti

3.3 SOAP JA WSDL

SOAP on yksinkertainen tiedonvälitykseen käytettävä protokolla. SOAP-viestit ovat aina XML-muotoisia, ja ne välitetään jonkun sopivaksi katsotun, toisen

tiedonvälitysprotokolla avulla. (Järvinen 2002, 3.) SOAP-määritys koostuu neljästä osasta: SOAP-kirjekuoren (envelope), tiedonkuvaustavan (encoding) ja RPC-etäkutsujen määrityksistä sekä HTTP-sidonnasta. SOAPin määrittelemät viestit ovat yhdensuuntaista tiedonsiirtoa, joista voidaan koota suurempia kokonaisuuksia, esimerkiksi pyyntö/vastauspareja. Yleisimmin SOAP-viestejä kuljetetaan HTTP:n päällä, ja niinpä SOAP määrittelee HTTP-sidonnan (HTTP-binding), joka kertoo, kuinka HTTP:tä tulisi käyttää SOAP-viestien vällittämiseen. (Järvinen 2002, 79.)

WSDL (Web Services Description Language) on Microsoftin ja IBM:N W3C:lle tekemä standardiehdotus, jonka W3C on ottanut käsittelyyn vuoden 2001 alussa. WSDL on XML:ään perustuva formaatti, jonka tarkoituksena on kuvata Web Serviceitä kahden päätepisteen välillä kulkevien viestien avulla. WSDL ei ota kantaa siihen, mitä viestit sisältävät, eikä siihen, mitä protokollaa tiedonvälitykseen käytetään. (Järvinen 2002, 57.)

WSDL on kieli, jota käytetään kuvaamaan jotakin tiettyä SOAP-palvelua. WSDL-dokumentti siis kertoo sitä ymmärtävälle ohjelmistolle, mistä jokin SOAP-palvelu löytyy ja mitä palveluja se tukee. Ilman tätä dokumenttia SOAP-palvelimia on vaikea paikallistaa, ja toisaalta sekä metodien nimet että parametrit jäävät arvoituksiksi. (Järvinen 2002, 3.)

Sovelluskehittäjän täytyy tietää, mistä jokin WSDL-dokumentti löytyy ohjelmallisesti. Tässä kohdin apuun tulee UDDI (Universal Description, Discovery, and Integration), joka on rajapinta, jolla voidaan hakea verkkopalveluita erilaisin kriteerein jostakin keskitetystä rekisteristä. (Järvinen 2002, 4.)

3.4 REST

HyperText Transfer Protocol (HTTP) on yleinen, tilaton protokolla, joka säätelee tiedostojen välitystä verkossa. Alun perin se kehitettiin Euroopan Laboratorion hiukkasfysiikkaan osastolle (CERN). Tietoa välitetään HTTP:n välityksellä dokumenttien muodossa, jotka tunnistetaan käyttämällä URIa (Uniform Resource Identifiers). Jokaisella resurssilla Webin yli on URL-osoite (Uniform Resource Locator), joka tunnistaa resurssin sijainnin ja kuvailee, kuinka sitä voi lähestyä. URL tarjoaa protokollan nimen, jolla pääsee käsiksi resurssiin, osoitteen (joka voi olla

domainin nimi tai IP-osoite) koneeseen, missä resurssi sijaitsee ja hierarkkisen kuvauksen resurssin sijainnista tuolla koneella. (Alonso ym. 2004, 95.)

HTTP-protokolla noudattaa perinteistä asiakas/palvelin-arkkitehtuuria, missä asiakasohjelmisto ymmärretään useimmiten WWW-selaimeksi ja palvelin HTTP-protokollaa ymmärtäväksi WWW-palvelimeksi. Asiakasohjelma lähettää palvelimelle pyynnön (request), johon palvelin vastaa (response). (Järvinen 2002, 54.)

Pyyntöviesti osoittaa todellisen operaation, joka halutaan toteuttaa palvelimella. Tyypillisiin metodeihin kuuluvat OPTIONS (lähettää tietoa viestintäkeinoista, joita tuetaan kyseisellä palvelimella), GET (hakee minkä tahansa dokumentin, joka on määritelty pyynnössä tai, jos pyyntö määrittelee ohjelman, hakee dokumentin, jonka ohjelma tekee), POST (lisää tietoa, jotka on liitettynä pyyntöön paikkaan, joka on määritelty), PUT (päivittää tiedot pyynnössä määriteltyyn paikkaa), DELETE (poistaa tiedon johon viitataan pyynnössä). (Alonso ym. 2004, 95.)

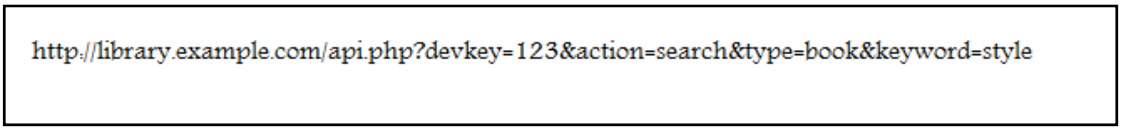
REST on tilaton menetelmä sovelluksille esittää pyyntöjä tarvittavaan palveluun. Jokaisella pyynnöllä on kaksi keskeistä osaa: päätepiste (yleensä URL) ja sanoma, joka ilmoittaa pyydetyn toiminnon. Se on samankaltainen toiminta kuin, jos Web-selain pyytää näyttämään jotain verkkosivua. Selain tietää päätepisteen (verkkosivun URL-osoite kyseessä) ja se tietää, minkä toiminnon se haluaa suorittaa (lataa kyseisen sivun). Lisäksi kuten web-selailussa, on olemassa menetelmiä joilla jäljitellä tilallista yhteyttä. Web-selailussa näitä kutsutaan istunnoiksi (sessions), REST:llä on samanlainen rakenne. (Reinheimer 2006, 99.)

Vaikka REST voi olla minkä tahansa reitin (ei vain HTTP / HTTPS) kautta yhteydessä sovellusrajapintaan, on HTTP loogisin valinta. REST pyyntöön kuuluu, että pyyntö lähetetään tiettyyn URL-osoitteeseen, vastauksena tulee esimerkiksi XML-dokumentti, joka sisältää palvelimen vastauksen. XML-dokumentti sitten jäsennetään, ja halutut tiedot otetaan haltuun. Jokaisella REST-pyynnöllä on muutamia yhteisiä elementtejä:

- Päätepiste URL : Täydellinen osoite haluttuun tiedostoon. REST Web Service voi mahdollisesti sisältää joko vain yhden tiedoston, joka käsittelee kaikki pyyntötyypit, tai eri tiedostoja eri pyyntötyyppejä varten.

- (Kehittäjän) tunnus: Useimmat REST palvelut edellyttävät jonkinlaista kehittäjä Id:tä tai avainta, joka lähetetään jokaisen pyynnön mukana. Tämä on keino tunnistaa hakemuksen lähettäjä ja sitä käytetään yleensä seurantaa varten. Jotkut Web Servicet voivat käyttää tätä arvoa rajaamaan kyselyiden määrää tietyn aikavälin aikana.
- Haluttu toiminto: Harvalla palvelimella on ainutlaatuinen päätepiste jokaiselle mahdolliselle pyynnölle. Siksi pyynnön mukaan on yleensä sisällyttävä haluttu toiminto.
- Parametrit: Muutamia muuttujia täytyy liittää pyynnön mukaan. Esimerkiksi jos haluttu toiminto on hakea tietoa, niin paremetrit voisivat olla type ja keywords asiasanoilla book ja style.

Näillä elementeillä voidaan rakentaa teoreettinen pyyntö, joka on esitelty kuvassa 13. (Reinheimer 2006, 99.)



KUVA 13. REST pyyntö (Reinheimer 2006, 99)

3.5 JSON

JSON (JavaScript Object Notation) on kevyt tiedonsiirtomuoto. Se pohjautuu JavaScript-koodin osajoukkoon (Standard ECMA-262 3rd Edition - December 1999). JSON on tekstiformaatti, joka on täysin ohjelmointikielestä riippumaton, mutta käyttää käytäntöjä, jotka ovat tuttuja C-perheen kieliä ohjelmoineille, kuten C, C + +, C #, Java, JavaScript, Perl, Python. Nämä ominaisuudet tekevät JSON:sta ihanteellisen tiedonsiirtomuodon. (Introducing JSON.) Alapuolella on esimerkki JSON-tietorakenteesta .

```
{
     "etunimi": "Joku",
     "sukunimi": "Esimerkki",
     "ika": 25,
 }
```

Alapuolella on esimerkki mahdollisesta vastaavasta tietorakenteesta XML-dokumenttina.

```
<Henkilo>
  <etunimi>Joku</etunimi>
```

```
  <sukunimi>Esimerkki</sukunimi>
  <ika>25</ika>
</Henkilo>
```

Sovellukseen täytyy valita välitettävien viestien tiedonsiirtomuoto ja viestien vaihtoprotokolla, kun halutaan kommunikoida etätietokoneen kanssa. Esimerkiksi SOAP-pohjaiset Web Servicet käyttävät XML-muotoista dataa, jonka tietosisältö on käärittyä SOAP-kirjekuoreen. XML toimii hyvin monissa sovelluksissa, mutta siinä on myös joitakin haittoja, jotka tekevät siitä vähemmän ihanteellisen toisissa. Esimerkiksi XML ei ole ihanteellinen vaihtoehto Ajax-tyyliin verkkosovelluksissa. Ajax on tekniikka, jota käytetään rakentamaan interaktiivisia verkkosovelluksia. Vaikka useimmat selaimet osaavat rakentaa, lähettää ja jäsentää XML-tietoa, niin JSON tarjoaa standardoidun tiedonsiirtomuodon, joka sopii paremmin juuri Ajax-verkkosovelluksiin. JSON standardi on kevyt ja sitä voidaan jäsentää JavaScript-toteutusten kanssa helposti, joten se on ihanteellinen tiedonsiirtomuoto Ajax-verkkosovelluksissa. JSON ei kuitenkaan rajoitu vain Ajax-verkkosovelluksiin, vaan sitä voidaan käyttää lähes missä tahansa tilanteessa, jossa sovellusten tarvitse vaihtaa tai tallentaa jäsenneltyä tietoa tekstinä. (Aziz & Mitchell 2007.)

JSON muotoista dataa voidaan käsitellä JavaScriptillä ilman ongelmia. Jos halutaan muuttaa JSON tekstimuotoista dataa objektirakenteiseksi, voidaan käyttää eval-funktiota, josta on esimerkki kuvassa 14. Teksti on käärittävä sulkuihin, jotta vältetään epäselvä JavaScript-koodi. (JSON in JavaScript.)

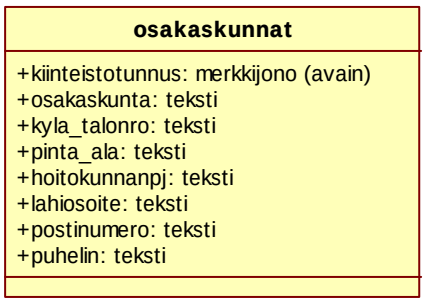
```
var myObject = eval('(' + myJSONtext + ')');
```

KUVA 14. eval-funktio JavaScript-koodissa

4 TOTEUTUS

Sain käsiini viisi PDF-dokumenttia ja Excel-tiedoston, josta löytyivät Puulan kalastusalueen osakaskuntien tiedot. Tarkoituksena oli luoda sovellus näistä PDF-dokumenteista, josta saisi selville osakaskuntien vesialueiden omistajien alueet. PDF-dokumenteissa oli osakaskuntien rajat ja Puulan kalastusalue piirretty Maanmittaustoimiston toimesta viideksi eri osa-alueeksi ja jokainen osa-alue toimi omana PDF-dokumenttinaan. Kalastusalueen osia ovat itäinen, pohjoinen, läntinen, eteläinen ja keskinen. Toimeksiantajan vaatimuksena oli, että PDF-dokumentissa osakaskunnan kohdalla tuli olla muistilappu ja sen sisältämän osakaskunnan tiedon tuli tulla jostain ulkopuolelta ja sen piti olla päivitettävissä. Selväksi tuli myös, että sovellus tulisi osaksi puula.fi-verkkosivustoa. Sain toimeksiantajalta Adobe Acrobat -ohjelmiston ja tietokoneen, joilla sovellus oli tarkoitus toteuttaa. Työnimikkeeksi projektille tuli tietokantapohjaisen karttasovelluksen suunnittelu ja toteutus.

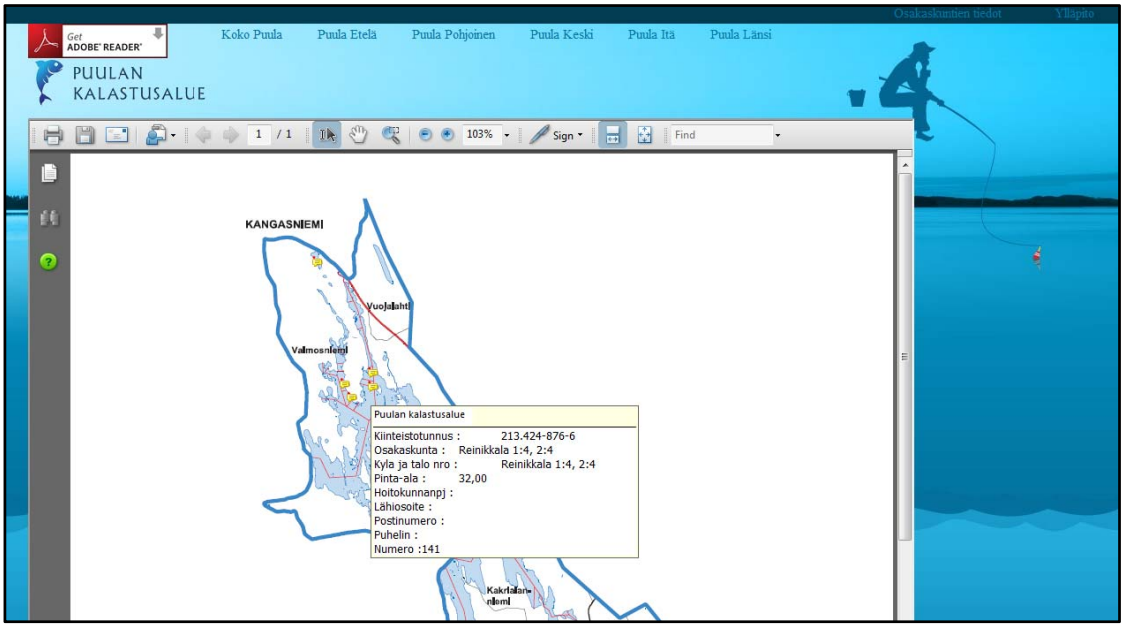
Aluksi Excel-dokumentista aloin heti ajatella, että sovellus tarvitsee tietokannan, jonne osakaskuntien tiedot sijoitetaan. Tietokannalle piti tehdä myös ylläpitotyökalut. Ajattelin, että tietokanta ja palvelin osuus sovelluksesta tehdään PHP/MySQL-sovelluksena. Tämä vielä varmistui, kun selvitin, että puula.fi palvelimella oli jo php-tuki ja MySQL-tietokanta. Seuraavaksi suunnittelin ja toteutin tietokannan osakaskuntien tietoja varten, jonka rakenne selviää kuvasta 15. Seuraavaksi rakensin tietokannalle ylläpidon php-sovelluksella. Ylläpidossa on mahdollista muokata ja lisätä tietoja. Ylläpitoon pääsee karttasovelluksen kautta omasta linkistään, joka aukeaa uuteen ikkunaan. Ylläpitoon täytyy kirjautua syöttämällä tunnus ja salasana. Kirjautumisen toteutin sen vuoksi, ettei kuka tahansa pääse muokkaamaan osakaskuntien tietoja.



KUVA 15. Tietokanta

Myöhemmin sain tunnukset Ammattilaisen karttapaikkaan, josta jonkin osakaskunnan sijaintia voi selvitellä osakaskunnan kiinteistötunnuksen perusteella. Excel-dokumentissa oli jokaisen osakaskunnan kohdalla myös kiinteistötunnus, joten sijainnin selvittäminen oli mahdollista. Tosin osakaskunnilla on paljon pieniä alueita Puulan kalastusalueella, joten sijaintien määrittämisestä PDF-dokumentteihin tuli väkisinkin vähän yksinkertaistettua. Osakaskuntia Puulan kalastualueella excel-dokumentin mukaan on noin kaksisataa.

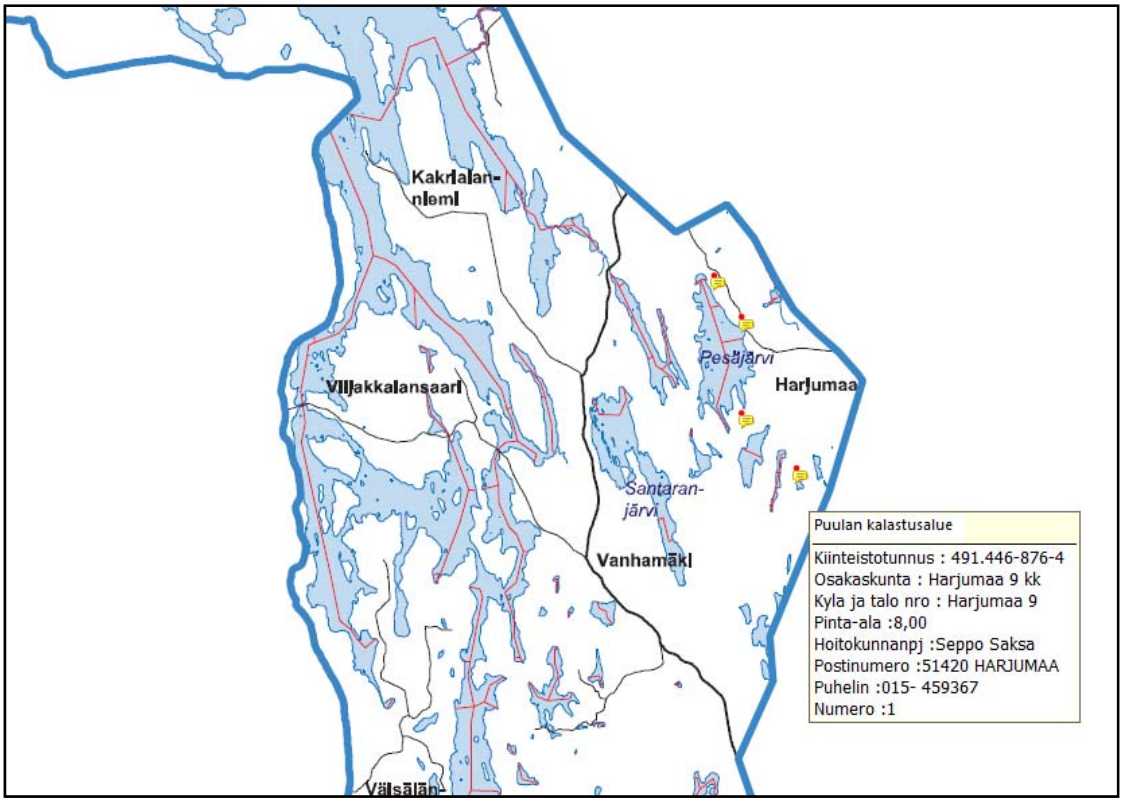
Tein karttasovellukselle omat verkkosivut, ja sijoitin kartat omille sivuilleen niille varatuille paikoilleen (iframe-kenttiin). PDF-karttoja on mahdollista katsoa myös omana sovelluksenaan, eikä ainoastaan verkkosivuilla. Verkkosivuilla kävijän on mahdollista myös selata osakaskuntien tietoja. Linkki osakaskuntien tietoihin on yläpalkin oikeassa reunassa ylläpito linkin vasemmalla puolella. Verkkosivujen toteutus näkyy kuvassa 16. Graafisen ilmeen otin suoraan puula.fi-sivuilta, jotta sovellus pysyisi samassa teemassa kuin pääsivuilla.



KUVA 16. Karttasovellus verkkosivuilla

Projektin seuraava vaihe oli todellista tiedon keräämistä, luin artikkeleita, www-sivuja, oppaita ja jo toteutettuja sovelluksia Acrobat maailmassa. Minun kokemukseni ennen tät projektia PDF.tiedostoista, oli se, kun olin joitain kirjoituksia tulostanut tai tallentanut PDF-tiedostoiksi. Tein monenlaisia virityksiä ja kokeilin kaikennäköisiä

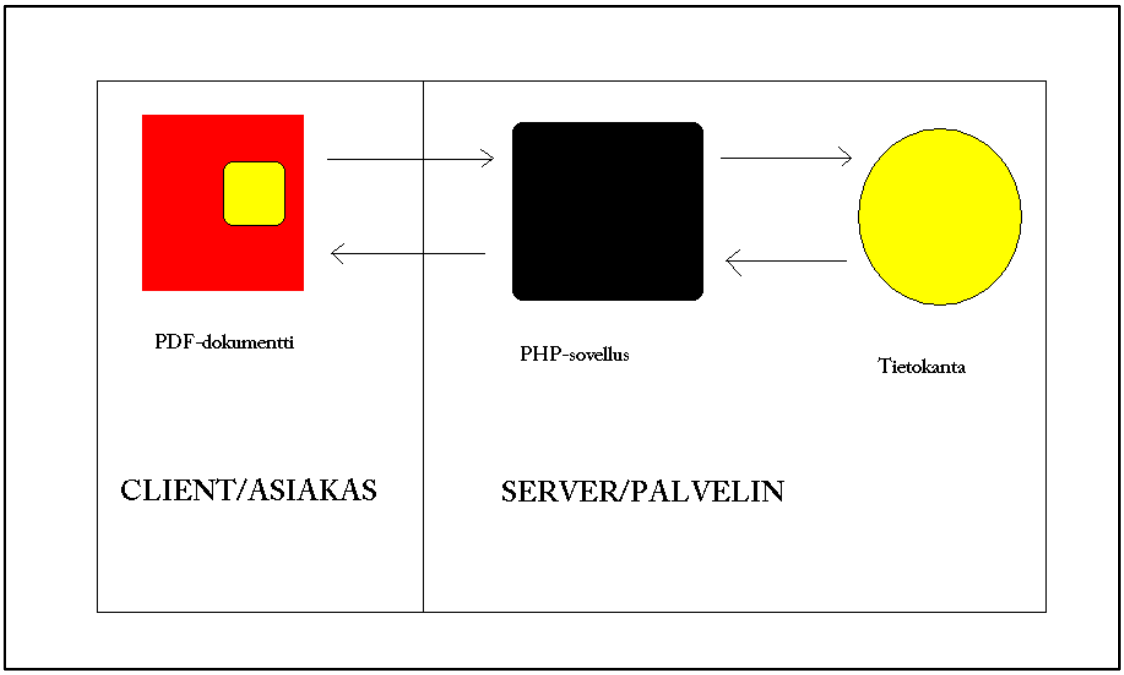
temppuja, jotta saisin tietoja tietokannasta PDF-dokumenttiin. Esimerkiksi loin linkkejä Acrobatin työkaluilla PDF-dokumenttiin ja ohjasin ne php-tiedostoon palvelimelle, joka näytti sitten haluttuja tietoja tietokannasta. Tässä kuitenkin oli ongelmana se, että kartalta ajauduttiin pois toiselle verkkosivustolle ja näin karttasovellus ei ollut helppokäyttöinen. Joten ainoa oikea vaihtoehto oli saada tiedot tulostumaan PDF-dokumentin muistilappuihin, kuten lopullisen toteutuksen esimerkissä kuvassa 17.



KUVA 17. PDF-kartta, jossa muistilappu näyttää tiedon Puulan kalastusalueen osakaskunnasta

Sain selville, että JavaScript-koodia voidaan sijoittaa PDF-dokumentteihin. Acrobatissa JavaScript-koodia on versionimellä AcroJS. Seuraavaksi sain vielä selville, että JavaScript-koodia voidaan ajaa PDF-dokumentin avautuessa eli koodia voidaan sijoittaa dokumenttitasolle. Acrobat Standardissa voidaan JavaScript-koodia sijoittaa linkkeihin ja nappeihin eli kenttätsolle. Kun halutaan sijoittaa JavaScriptiä muille kuin kenttätasolle, kuten esimerkiksi dokumenttitasolle, niin vaaditaan Acrobat Pro –ohjelmisto. Seuraavaksi selvitin miten JavaScriptillä PDF-dokumentissa otetaan eri tekniikoin yhteys palvelimelle ja tietokantaan. Acrobat tarjoaa SOAP-yhteyden ja WSDL-tiedoston käytön mahdollisuuden, mutta tämän käyttäminen tuntui liian

raskaalta vaihtoehdolta kyseiseen sovellukseen. Valitsin yksinkertaisemman REST-tekniikan, joka lähettää palvelimelle GET-metodilla pyynnön ja parametrina pyynnöllä toimii halutun tiedon numero. Viestimuodoksi valitsin kevyemmän JSONin SOAPiin kuuluvan XML:n sijasta. Kuvassa 18 näkyy lopullinen sovellusarkkitehtuuri. Siinä kuvataan asiakkaana toimivan PDF-tiedoston ja palvelinpuolen php-sovelluksen lopullinen yhteistyö.



KUVA 18. Sovelluksen arkkitehtuuri

Sovelluksessa viiteen PDF-dokumentiin on sijoitettu JavaScript-koodia dokumenttitasolle. JavaScriptin tarkoituksena on muodostaa muistilappuja PDF-dokumenttiin, hakea niiden sijaintitiedot ja tietosisältö palvelimen kautta tietokannasta. JavaScript lähettää pyynnön (GET) palvelimella toimivalle php-sovellukselle. Pyynnön parametrina toimii halutun tiedon numero ja näin php-sovellus osaa hakea oikean tiedon tietokannasta. Php-sovellus muuttaa seuraavaksi tietokannasta haetun viestisisällön JSON-muotoiseksi ja palauttaa sen PDF-dokumenttiin. Seuraavaksi PDF-dokumentin dokumenttitasolle sijoitetulla JavaScriptillä otetaan tiedot vastaan ja parsitaan ne eval-funktiolla paremmin käsiteltävään muotoon. Seuraavaksi halutut tiedot voidaan sijoittaa tulostumaan muistilapulle.

5 PÄÄTÄNTÖ

Tavoitteena opinnäytetyöllä oli tutkia PDF-dokumenttien soveltuvuutta verkkopalvelutekniikoihin projektin vaatimalla tavalla. Tarkemmin sanottuna tavoitteena oli tutkia, kuinka PDF-dokumentteihin saa luotua dynaamista sisältöä.

Lisäksi itselläni oli tavoitteena tietenkin oppia uusia asioita ja soveltaa niitä käytäntöön. Tällä kyseisellä opinnäytetyölläni täytin tavoitteeni kirkkaasti. Opinnäytetyötä oli mukava tehdä ja siitä sain lisää varmuutta siihen, että pystyn ratkaisemaan aluksi haastaviltakin tuntuvia ongelmia

PDF-tiedostojen maailma ennen tätä opinnäytetyötä oli minun osalta rajoittunut joidenkin Word-dokumenttien kääntämiseen PDF-dokumenteiksi, joten kerralla tuli todella paljon uutta tietoa ja opittavaa. Lisäksi kun työhön liitettiin erilaisten Web Service -tekniikoiden tutkiminen, niin se antoi lisäsyvyyttä aiheelle.

Toteuttamani sovellus Puulan kalastusalueelle toimii, kuten toimeksiantajan vaatimus oli. Käyttäjä pystyy muokkaamaan osakaskuntien tietoja muokkausta varten rakennetun ylläpidon kautta ja sitä kautta tiedot päivittyvät muistilappuihin PDF-dokumentissa.

Lopullisessa sovelluksessa on olemassa mahdollinen tietoturvauhka, joka liittyy PDF-dokumentin JavaScript-koodiin. Lehtiartikkelien mukaan Acrobatin JavaScript on tällä hetkellä verkkorikollisten suosima hyökkäyskohde, kun se tarjoaa tietoturva-aukon. Adobe on vastannut uhkaan siten, että he eivät tule ainakaan vielä tekemään asialle mitään. Ajaako tämä uhka sitten kokonaan alas JavaScriptin mahdollisuuden PDF-dokumenteissa? Sen tulevaisuus näyttää.

LÄHTEET

Tarvainen, Juha 2006. Acrobat 7. Docendo.

Adobe 2009. Adobe Acrobat Standard 9.0 Help. PDF- dokumentti. http://help.adobe.com/fi_FI/Acrobat/9.0/Standard/acrobat_standard_9.0_help.pdf. Luettu 12.4.2010.

Wraight, Dave 2004. Developing with Acrobat Javasript. Planet PDF. WWW-dokumentti. http://www.planetpdf.com/developer/article.asp?ContentID=6622. Luettu 13.4.2010.

Tähtinen, Sami 2005. Järjestelmäintegraatio. Gummerus.

Crockford, Douglas 2009. Introduction JSON.. WWW-dokumentti. http://www.json.org/example.html. Kirjoitettu 28.5.2009. Luettu 23.4.2010.

Nykänen, Ossi 2006. XML Toolkit. Docendo.

Reinheimer, Paul 2006. Professional Web APIs with PHP. Wiley Publishing.

Adobe 2006. Developing Acrobat Applications Using JavaScript. PDF-dokumentti. http://www.adobe.com/devnet/acrobat/pdfs/js_developer_guide.pdf. Luettu 24.4.2010.

Alonso, Gustavo, Casati, Fabio, Kuno, Harumi, Machiraju, Vijay 2004. Web Services. Springer.

Erl, Thomas 2004. Service-Oriented Architecture. Prentice Hall PTR.
Smith, Dori, Negrino, Tom 2007. JavaScript rakenna dynaamisia verkkosivuja. Readme.fi.

Korpela, Jukka 2006. PDF-muotoisten dokumenttien lukeminen (ja vähän niiden tekemisestäkin). WWW-dokumentti. http://www.cs.tut.fi/~jkorpela/pdf.html. Päivitetty 17.12.2006. Luettu 25.4.2010.

Barry, Douglas 2010. Web Services explained. WWW-dokumentti. http://www.service-architecture.com/webservices/articles/web_services_explained.html. Luettu 25.4.2010.

Barry, Douglas 2010. Service-oriented architecture (SOA) definition. WWW-dokumentti. http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html. Luettu 25.4.2010.

Booth, David 2003. Web Services Architecture. WWW-dokumentti. http://www.w3.org/TR/2003/WD-ws-arch-20030808/. Luettu 25.4.2010.

JSON in JavaScript 2010. json.org. WWW-dokumentti. http://www.json.org/js.html. Luettu 26.4.2010.

Introducing JSON 2010. json.org. WWW-dokumentti. http://www.json.org/. Luettu 26.4.2010.

Jarvinen, Jani 2002. Hajautetut verkkopalvelut. Docendo.

Aziz, Atif & Mitchell, Scott 2007. An introduction JavaScript Object notation (JSON) in JavaScript and .NET . WWW-dokumentti. http://msdn.microsoft.com/en-us/library/bb299886.aspx. Luettu 25.4.2010.